BAB V

PENGUJIAN DAN ANALISIS

Tujuan pengujian sistem ini adalah untuk menentukan apakah alat yang telah dibuat berfungsi dengan baik dan sesuai dengan perancangan. Pengujian pada sistem ini meliputi pengujian setiap blok maupun pengujian secara keseluruhan. Pengujian setiap blok ini dilakukan untuk menemukan letak kesalahan dan mempermudah analisis pada sistem apabila alat tidak bekerja sesuai dengan perancangan. Pengujian dibagi menjadi beberapa bagian, yaitu :

1. Pengujian *Driver* Motor.
2. Pengujian sensor *rotary encoder*.
3. Pengujian kecepatan Motor DC.
4. Pengujian sistem keseluruhan.

5.1 Pengujian *Driver* Motor DC

a. Tujuan

Pengujian *driver* motor *Direct Current* (DC) ini bertujuan untuk mengetahui *output driver* motor yang dibandingkan dengan masukannya yang kemudian dapat diketahui juga hubungan keluaran *Pulse Width Modulation* (PWM) dengan tegangan yang dibutuhkan untuk motor *Direct Current* (DC).

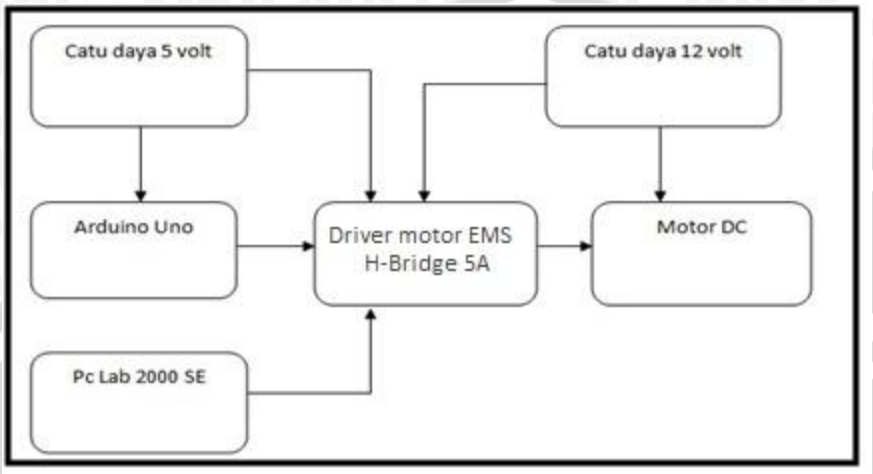
Peralatan yang digunakan

- Modul Arduino Uno
- *Driver* motor
- Motor DC
- Catu daya 5V dan 12V
- *Hardware* dan *software Pulse Width Modulation* (PWM)

b. Langkah pengujian

1. Merangkai peralatan seperti Gambar 5.1.
2. Mengunduh program yang mengeluarkan data berupa *Pulse Width Modulation* (PWM) dengan keluaran 10-250 dengan interval setiap 10 *Pulse Width Modulation* (PWM) pada *software* arduino ERW 1.0.5.

3. Jalankan *software* PC Lab 2000SE.
4. Pilih mode osiloskop lalu pilih *Run* untuk menjalankan osiloskop.
5. Mengamati sinyal kontrol berupa *duty cycle* pada osiloskop untuk setiap pengujian mikrokontroler dan *driver*.



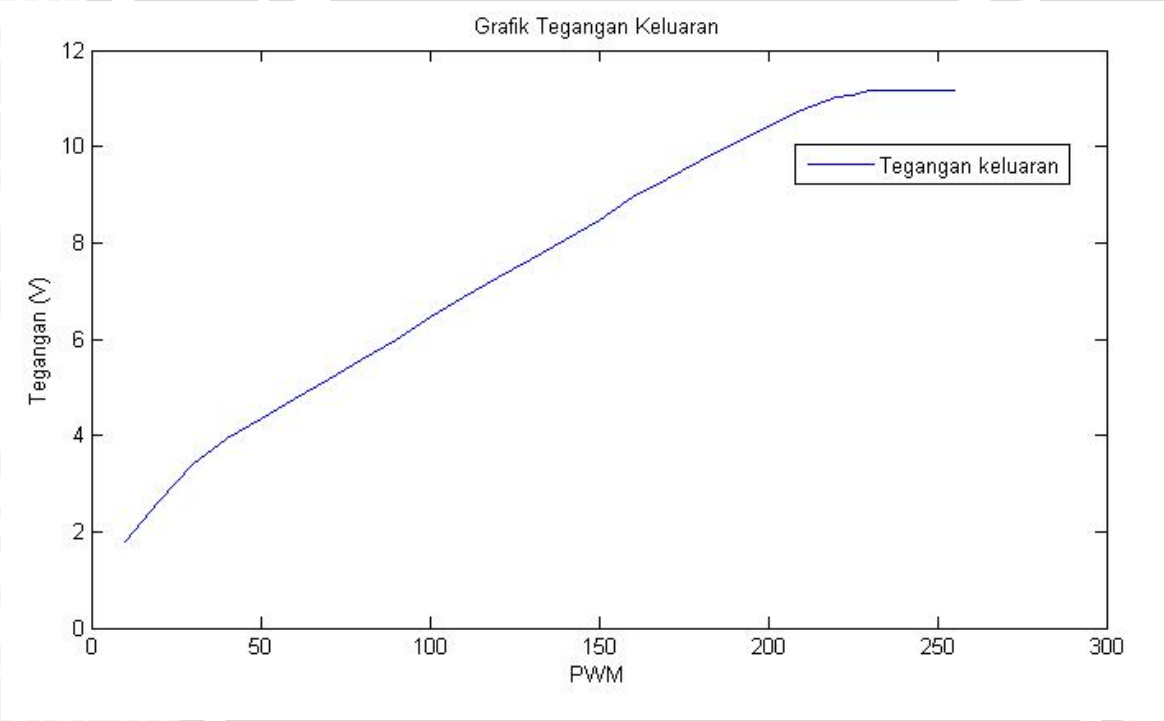
Gambar 5.1 Diagram Blok Pengujian *Driver* Motor DC

c. Hasil pengujian

Tabel 5.1 Hasil Pengujian *Driver* Motor DC

PWM	*Duty Cycle Driver*	Tegangan *Driver* (V)
10	11	1.78
20	19.3	2.63
30	26.4	3.38
40	31.3	3.95
50	35.1	4.35
60	38.9	4.77
70	42.6	5.18
80	64.5	5.58
90	50.2	5.99
100	54.6	6.46
110	58.6	6.87
120	62.3	7.27
130	65.4	7.67
140	69.2	8.07

150	73	8.47
160	77.6	8.94
170	81.4	9.33
180	84.7	9.71
190	88.1	10.08
200	91.4	10.43
210	93.9	10.74
220	95.6	11.02
225	97.7	11.07
230	??	11.15
240	??	11.16
250	??	11.17
255	??	11.18



Gambar 5.2 Grafik Hubungan Tegangan dengan PWM

Berdasarkan hasil pengujian pada Tabel 5.1 dan grafik Gambar 5.2 dapat diketahui bahwa tegangan yang keluar berubah-ubah sesuai dengan perubahan *Pulse Width Modulation (*PWM*)* dengan demikian dapat disimpulkan *driver* dapat berfungsi dengan baik.

5.2 Pengujian Kecepatan Motor DC

a. Tujuan

Untuk mengetahui kecepatan motor yang diberikan masukan berupa *Pulse Width Modulation* (PWM) yang berbeda-beda.

b. Peralatan yang digunakan

- Catu daya 5 volt dan 12 volt
- Modul Arduino Uno
- *Driver* motor
- Motor DC
- Sensor *rotary encoder*

c. Langkah pengujian

Mengukur kecepatan motor dengan nilai *Pulse Width Modulation* (PWM) yang berbeda, sehingga didapatkan kecepatan yang diinginkan.

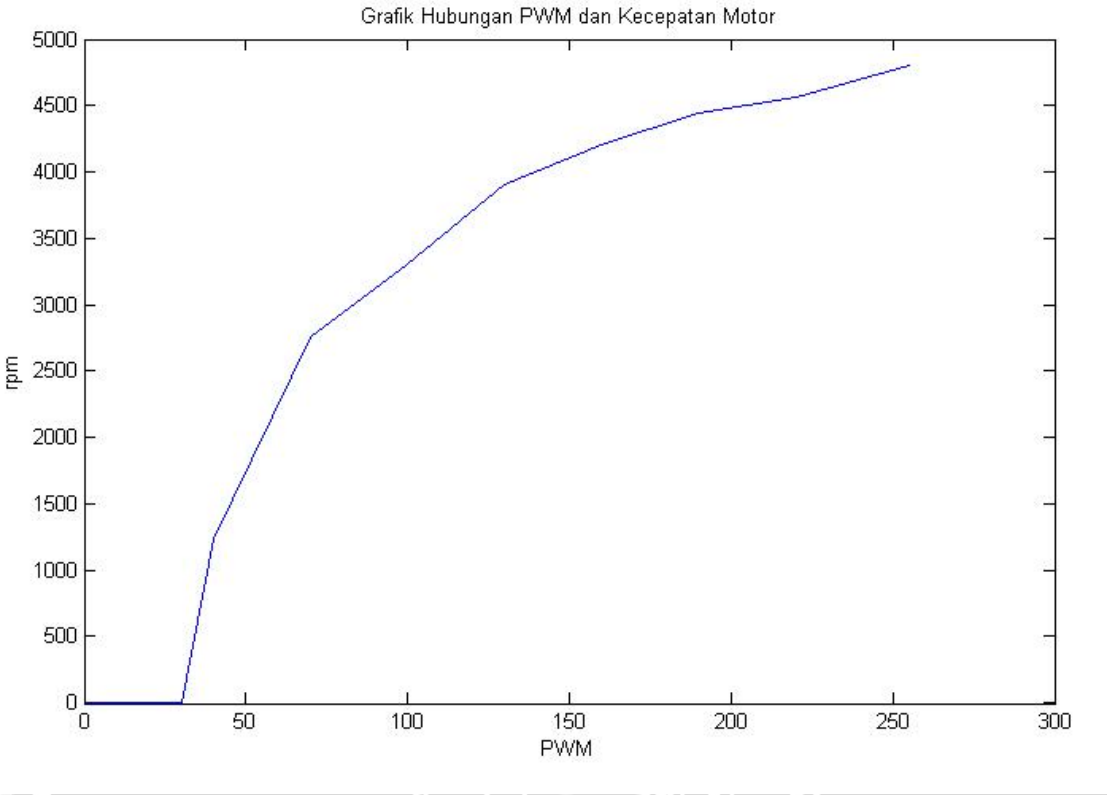
d. Hasil pengujian

Hasil pengujian kecepatan motor dengan nilai *Pulse Width Modulation* (PWM) yang ditentukan ditunjukkan pada Tabel 5.2

Tabel 5.2 Hasil Pengujian Pembacaan Sensor Motor

PWM	rpm
0	0
10	0
20	0
30	0
40	1240
70	2760
100	3300
130	3900
160	4200
190	4440

220	4560
255	4800



Gambar 5.3 Grafik Hubungan PWM dengan Kecepatan Motor.

Berdasarkan hasil pengujian pada Tabel 5.2 dan grafik Gambar 5.3 dapat dilihat motor mulai berputar di kisaran *Pulse Width Modulation* (PWM) 40, kecepatan motor mengalami perubahan yang besar dalam kisaran *Pulse Width Modulation* (PWM) 40 – 150 sedangkan pada *Pulse Width Modulation* (PWM) 150 - 255 perubahan kecepatan tidak begitu besar sehingga dapat disimpulkan motor berputar dengan masukan berupa *Pulse Width Modulation* (PWM) yang berbeda.

5.3 Pengujian Keseluruhan

a. Tujuan

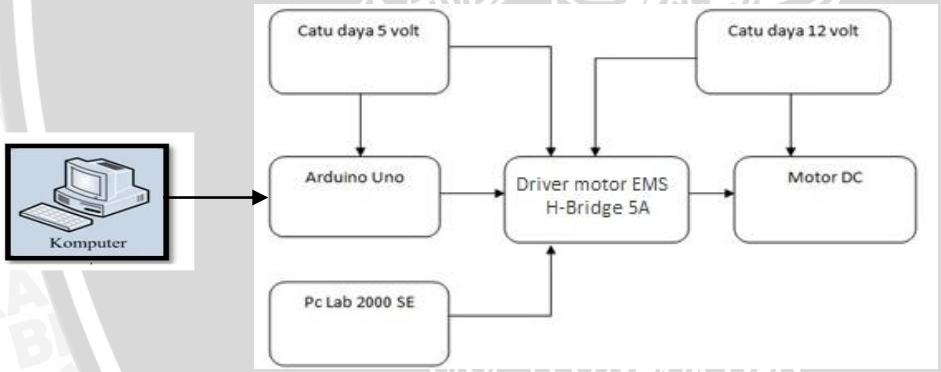
Pengujian ini dilakukan untuk mengetahui bagaimanakah karakteristik dari plant motor DC. Kemudian dicari fungsi alih dari plant dan kontrolernya untuk selanjutnya di simulasikan menggunkan *software* MATLAB.

b. Peralatan yang digunakan

- Motor DC dan sensor *rotary encoder*
- Catu daya 12 volt dan 5 volt
- Modul Arduino Uno
- *Driver* motor menggunakan modul EMS 5A *H-Bridge*
- Kabel serial

c. Langkah Pengujian

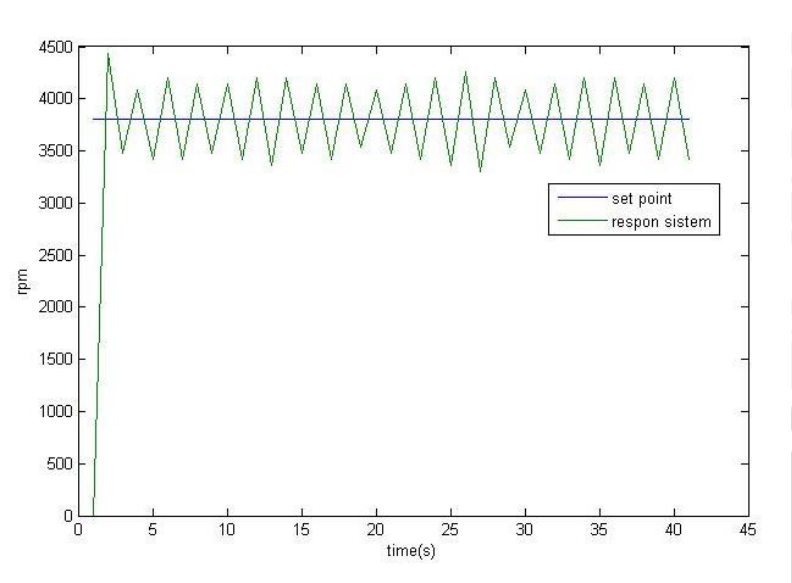
1. Menghubungkan rangkaian seperti Gambar 5.4.
2. Mengunduh program pembangkit sinyal PRBS sesuai dengan nilai PWM tertentu melalui *software* Arduino ERW 1.0.5.
3. Menampilkan dan merekam nilai rpm motor DC pada serial monitor *software* Arduino ERW 1.0.5.
4. Membuat grafik dari data yang didapat
5. Mengolah data-data yang didapat menggunakan *software* MATLAB R2013a



Gambar 5.4 Diagram Blok Pengujian Keseluruhan Sistem

d. Hasil pengujian

Pengujian dilakukan dengan membandingkan respon sistem dari kontroler PID metode Ziegler-Nichols 2 dengan *self tuning* PID kontroler metode *backward rectangular*. Langkah pertama adalah dengan melihat respon motor kontroler PID Ziegler-Nichols 2 setelah diberi kenaikan Kcr = 2 seperti dalam Gambar 5.5.



Gambar 5.5 Respon PID Zygler Nichols

Berdasarkan Gambar 5.5, didapatkan nilai Pcr sebesar 2. Nilai parameter kontroler PID ditentukan berdasarkan tabel aturan dasar Ziegler-Nichols dengan *critical gain* Kcr dan *critical period* Pcr yang ditunjukkan dalam Tabel 5.3.

Tabel 5.3 Dasar Ziegler-Nichols Berdasarkan *Critical Gain* Kcr dan *Critical Period* Pcr e. (Ogata K., 1997)

TipeKontroler	K_p	T_i	T_d
P	$0.5\ Kcr$	∞	0
PI	$0.45\ Kcr$	$\frac{1}{1.2} Pcr$	0
PID	$0.6\ Kcr$	$0.5\ Pcr$	$0.125\ Pcr$

maka nilai parameter PID yang diperoleh adalah :

Kp = 0.6 x 2 = 1.2

Ti = 0.5 x 2 = 1

Td = 0.125 x 1 = 0.25

Dengan demikian dapat ditentukan nilai Kp, Ki, dan Kd adalah:

Kp = 1.2

Ki = $\frac{K_p}{T_i}$

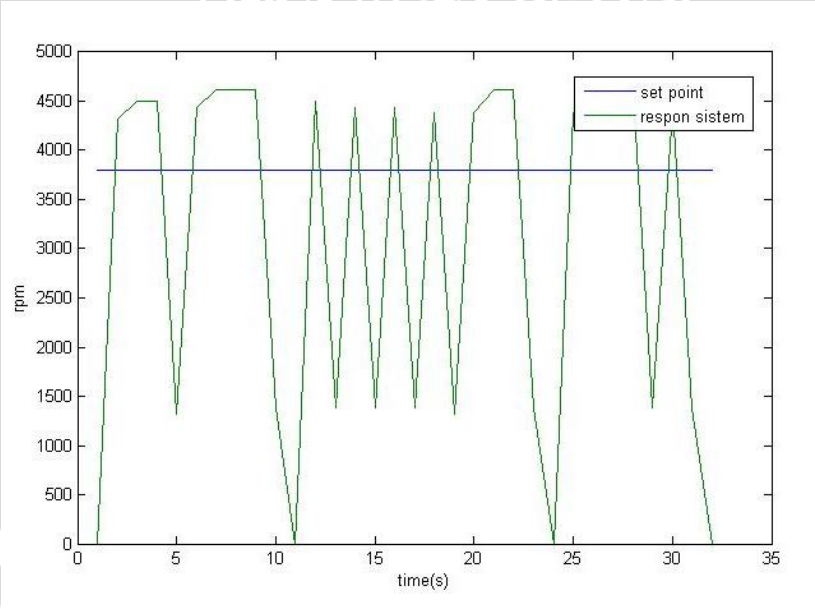
= 1.2/1

= 1.2

Kd = Kp x Td

= 1.2 x 0.25 = 0.3

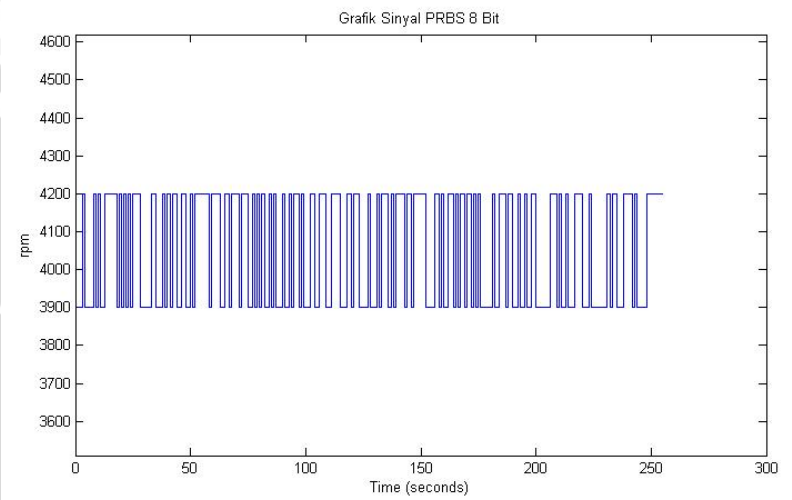
Dari penghitungan penentuan nilai penguatan dari metode kedua Ziegler-Nichols di atas diperoleh Kp = 21, Ki = 10,5, Kd = 10,5 yang akan digunakan untuk pengendali kecepatan motor DC. Dengan nilai Kp, Ki, dan Kd yang sudah diperoleh, maka grafik sistem respon yang diperoleh ditunjukkan oleh Gambar 5.6.



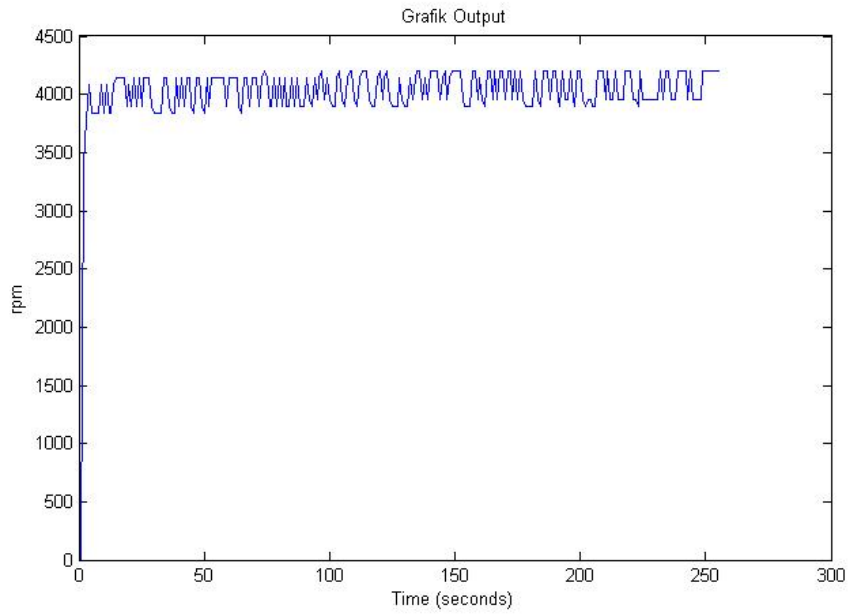
Gambar 5.6 Respon Sistem dengan Kp, Ki, dan Kd Berdasarkan Ziegler-Nichols

Dari grafik Gambar 5.6 bisa dilihat sistem tidak dapat menggunakan perhitungan Ziegler-Nichols tidak diperoleh respon osilasi yang divergen sebagaimana yang di kriteriakan *o*leh ziegler -nichols tipe 2. Untuk mendapatkan respon sistem yang lebih baik, maka dilakukan pengujian dengan menggunakan kontroler *self tuning* PID.

Langkah pertama, proses pengambilan data *input* dimulai dengan membangkitkan sinyal uji PRBS 8bit menggunakan metode ARX derajat 2 seperti yang ditunjukkan pada Gambar 5.7 dan Gambar 5.8

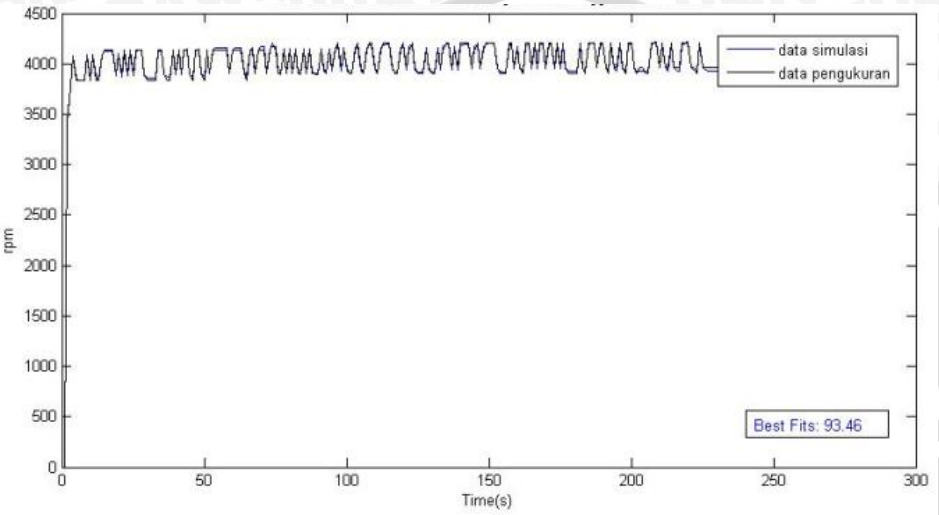


Gambar 5.7 Sinyal Uji PRBS 8 Bit



Gambar 5.8 Respon *Plant* Hasil Dari Uji Sinyal PRBS 8 Bit

Pada Gambar 5.9 menunjukan nilai *Best Fit* yang dicapai oleh kisaran sinyal PWM 160-190 sebesar 93,46. Nilai Best Fit ini merupakan nilai Best Fit terbaik dibandingkan pengujian pada kisaran sinyal PWM yang lain, sehingga range nilai PWM ini akan dipilih untuk digunakan pada pengujian selanjutnya.



Gambar 5.9 Best Fit Sinyal Uji PRBS Sebesar 93,46

Dari pengujian Best Fit diatas diperoleh nilai a_1, a_2, b_1, b_2 sebagai berikut:

a_1 = -0,6854 a_2 = 0,04321 b_1 = 0,6565 b_2 = -0,2984

Sehingga fungsi alih plantnya adalah:

$$G(z^{-1}) = \frac{0{,}6565\,z^{-1} - 0{,}2984\,z^{-2}}{1 - 0{,}6854\,z^{-1} + 0{,}04321\,z^{-2}} \quad (5.1)$$

Langkah berikutnya setelah mendapatkan nilai a_1, a_2, b_1, b_2 adalah untuk memperoleh nilai K_P, T_I, T_D. Nilai-nilai tersebut adalah:

- $Kc = \frac{1-a_2}{b_2}$ (5.2)

$$= \frac{1 - 0{,}04321}{-0{,}2984}$$

$$= -3{,}2064$$

- $\alpha = \frac{a_1 + Kc\,.\,b_1}{-2}$ (5.3)

$$= \frac{-0{,}6854 + (-3{,}2064\ .\ 0{,}6565)}{-2}$$

$$= 1{,}3952$$

- $Kr = \frac{1-a_1+a_2}{b_1-b_2}$ (5.4)

$$= \frac{1 + 0{,}6854 + 0{,}04321}{0{,}6565 + 0{,}2984}$$

$$= 1{,}8103$$

- $f = -(a_1 + Kr\ .\ b_1 - 1)$ (5.5)

$$= -(-0{,}6853 + 1{,}8103\ .0{,}6565 - 1)$$

$$= 0{,}497$$

Dari nilai diatas, persyaratan yang terpenuhi adalah pada point dua, sehingga nilai Kpu dan Tu adalah

- $Kpu = Kr = 1{,}8103$ (5.6)
- $Tu = T_0\ .2 = 0{,}01\ .\ 2 = 0{,}02$ (5.7)

Maka nilai K_P, T_I, dan T_D adalah

- $Kp = 0{,}6\ .\ Kpu = 0{,}6\ .\ 1{,}8103 = 1{,}0862$ (5.8)
- $T_I = \frac{Tu}{2} = \frac{0{,}02}{2} = 0{,}01$ (5.9)
- $T_D = \frac{Tu}{8} = \frac{0{,}02}{8} = 0{,}0025$ (5.10)

Setelah mendapatkan nilai K_P, T_I, dan T_D, maka diperoleh nilai q_0, q_1, q_2, p_1 dan p_2 sebagai berikut:

- $q_0 = K_p\left(1 + \frac{T_0}{T_I} + \frac{T_D}{T_0}\right)$ (5.11)

$$= 1{,}0862\left(1 + \frac{0{,}01}{0{,}01} + \frac{0{,}0025}{0{,}01}\right)$$

$$= 2{,}4439$$

- $q_1 = -K_p\left(1 + 2\frac{T_D}{T_0}\right)$ (5.12)

$$= -1{,}0862\left(1 + 2\frac{0{,}0025}{0{,}01}\right)$$

$$= -1{,}6293$$

- $q_2 = K_p\frac{T_D}{T_0}$ (5.13)

$$= 1{,}0862\frac{0{,}0025}{0{,}01}$$

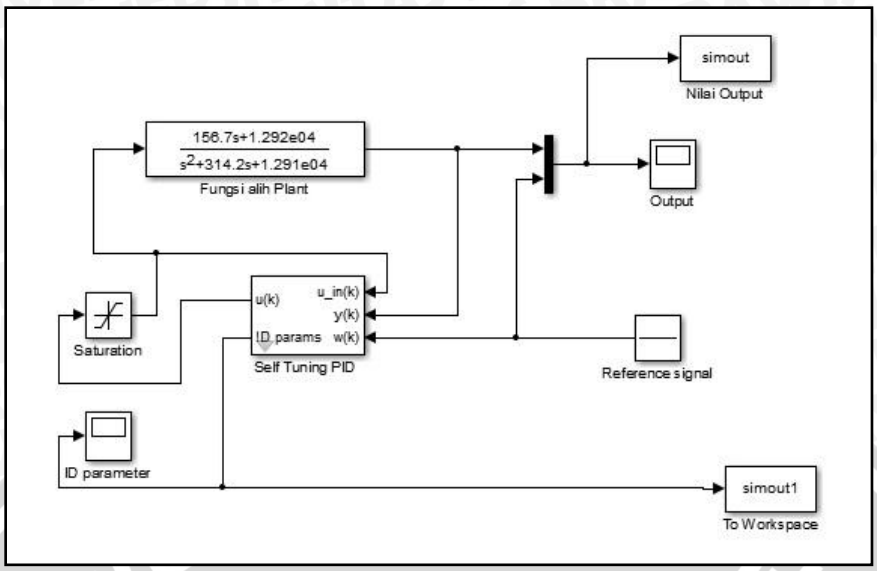
$$= 0{,}2715$$

- $p_1 = -1$ (5.14)
- $p_0 = 0$ (5.15)

Fungsi alih kontroler berdasarkan nilai-nilai diatas adalah

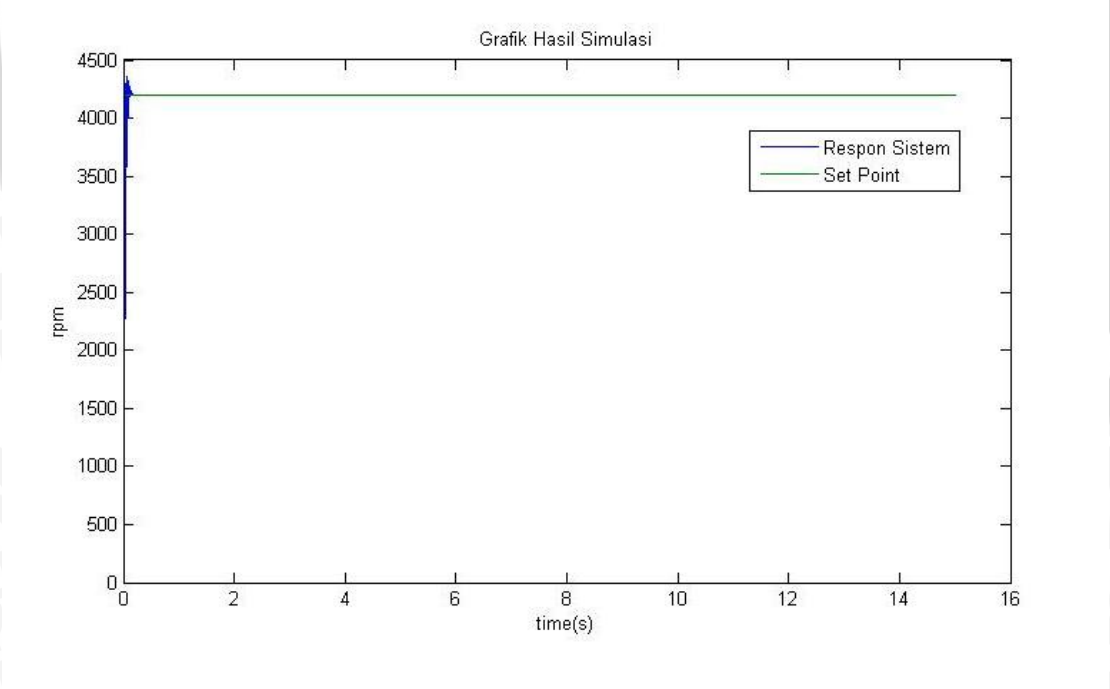
$$u(z) = \frac{2.4439 - 1.6293z^{-1} + 0.2715z^{-2}}{1 - z^{-1}} \quad (5.16)$$

Setelah fungsi alih kontroler dan fungsi alih plant diperoleh, langkah selanjutnya adalah mensimulasikan menggunakan *software* MATLAB. Seperti yang ditunjukkan pada Gambar 5.10



Gambar 5.10 Blok Diagram Simulink Sistem

Hasil dari simulasi ditunjukkan pada Gambar 5.11 yang menunjukkan grafik hasil simulasi untuk nilai *set point* tunggal sebesar 4200 rpm.



Gambar 5.11 Grafik Hasil Simulasi Pada *Set point* 4200 rpm

Dari grafik Gambar 5.9,diketahui bahwa hasil respon memiliki % *error* sebagai berikut:

$$\% \, Ess = \frac{|Average \;\; speed \;\; steady - \text{setpoint} \;|}{\text{setpoint}} \text{ x } 100\%$$

$$= \frac{|4200 - 4200|}{4200} \text{ x } 100 \%$$

$$= 0\%$$

Dari grafik di atas, terjadi *overshoot* dengan nilai sebesar:

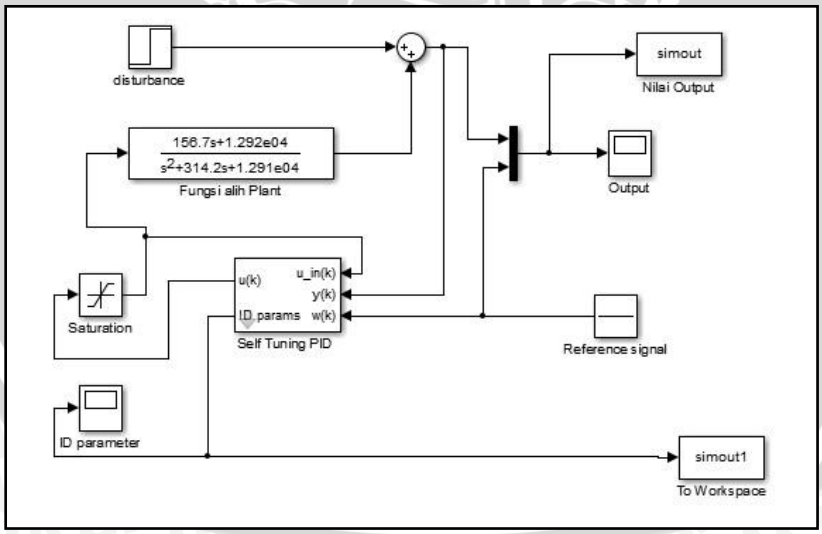
$$\% \text{ Mp} = \frac{|\text{RPMpuncak} \;\; - \text{RPM } steady \;|}{\text{RPM } steady} \text{ x } 100 \%$$

$$= \frac{|4361 - 4200|}{4200} \text{ x } 100 \%$$

$$= 3{,}833\%$$

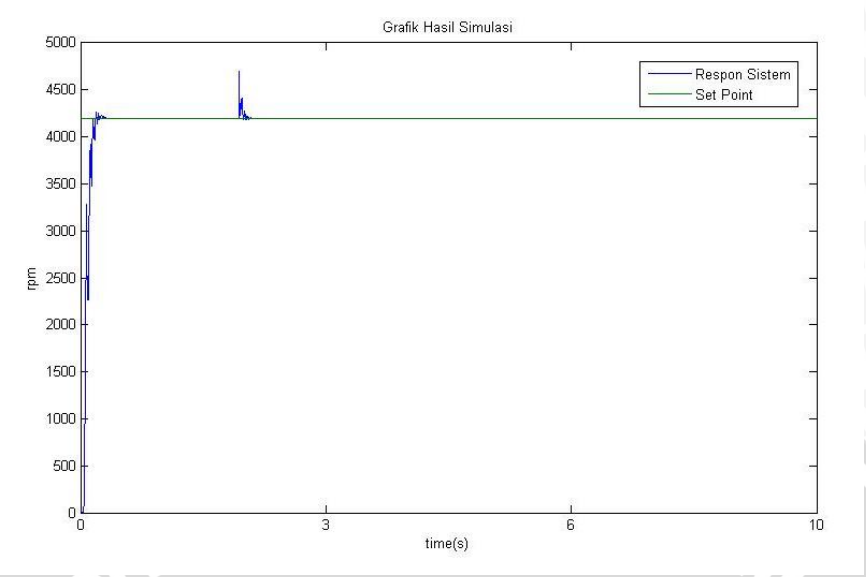
Dari hasil penghitungan didapatkan *error* sebesar 0% dan overshoot sebesar 3,83%.

Sebagai pembanding hasil dari respon sistem diatas, maka dilakukan simulasi dengan diberi *disturbance* seperti yang ditunjukkan pada Gambar 5.12



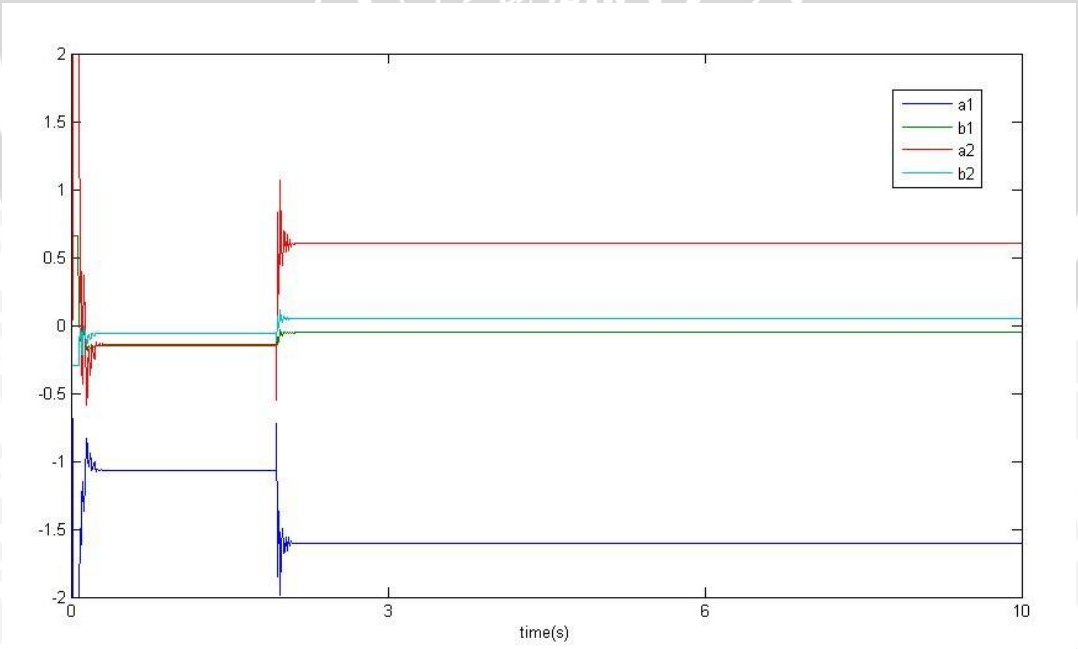
Gambar 5.12 Blok Diagram Simulink Sistem Diberi *Disturbance*

Simulasi dilakukan dengan cara memberikan *disturbance* konstan dalam rentang waktu 2 sampai 10 detik pada sistem *output*. Hasil simulasi ditunjukkan pada Gambar 5.13



Gambar 5.13 Grafik Hasil Simulasi Saat Diberi *Disturbance* Pada *Set point* 4200 rpm

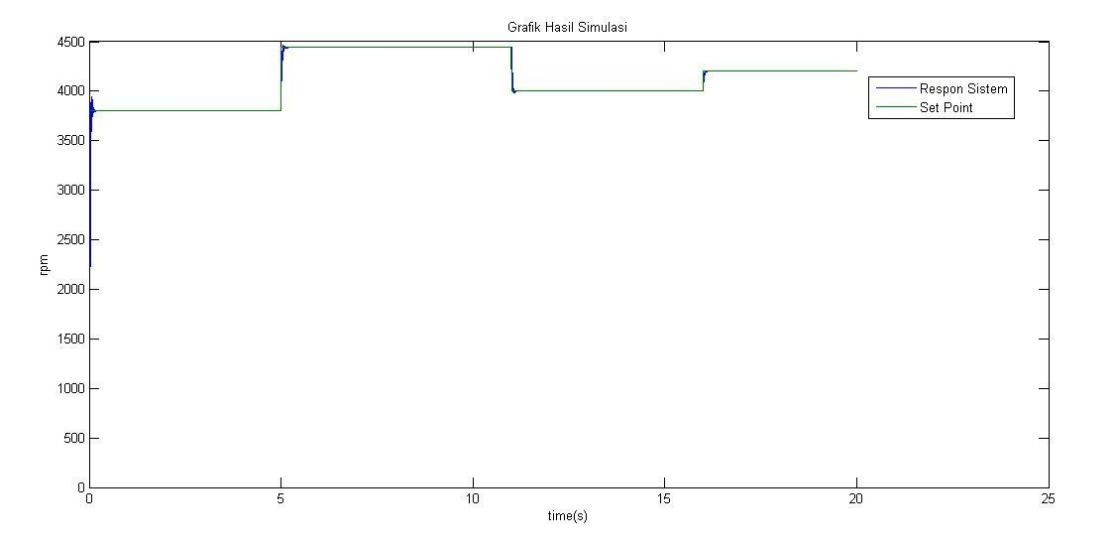
Untuk memperjelas cara kerja proses *self tuning* PID dalam proses kontrol menggunakan simulink MATLAB, kita bisa melihat adanya perubahan nilai a_1, a_2, b_1, b_2 pada *self tuning* PID apabila sistem diberi *disturbance*. Adapun perubahan nilainya ditunjukkan pada Gambar 5.14



Gambar 5.14 Pergerakan Parameter Kontrol Saat Sistem Diberi *Disturbance*

Dari Gambar 5.14 bisa diketahui bahwa apabila sistem diberikan *disturbance* dalam keadaan *steady,* maka kontroler secara otomatis juga akan melakukan perubahan *tuning* parameter a_1, a_2, b_1, b_2 sesuai kebutuhan. Saat sistem diberi *disturbance*, nilai a_1 berubah dari -1,06434 menjadi -1,5831, dimana untuk mencapai nilai 1,5831 dibutuhkan pengambilan data sebanyak 190 data dalam waktu 1,5 detik. Nilai a_2 berubah dari -0,1412 menjadi 0,581, dimana untuk mencapai nilai 0,581 dibutuhkan pengambilan data sebanyak 170 data dalam waktu 1,36 detik. Nilai b_1 berubah dari -0,14885 menjadi -0,05294, dimana untuk mencapai nilai -0,05294 dibutuhkan pengambilan data sebanyak 150 data dalam waktu 1,2 detik, dan nilai b_2 berubah dari -0,05685 menjadi 0,05058 dimana untuk mencapai nilai -0,05058 dibutuhkan pengambilan data sebanyak 120 data dalam waktu 1 detik.

Untuk mengetahui kehandalan dari nilai *tuning* sistem, maka dilakukan uji *set point* seperti yang ditunjukkan pada Gambar 5.15



Gambar 5.15 Grafik Hasil Simulasi Pada *Set point* 3800 rpm, 4440 rpm, 4000 rpm dan 4200 rpm

Dari grafik Gambar 5.15 diketahui bahwa sistem mengalami kenaikan *set point* dari 3800 rpm ke 4440 rpm, lalu turun dari 4440 rpm ke 4000 rpm, dan naik kembali dari 4000 rpm ke 4200 rpm.

Pada saat *set point* 3800 rpm, respon memili % *error* sebagai berikut:

$$\%\,Ess = \frac{|Average\ \ speed\ \ steady - \text{setpoint}\ |}{\text{setpoint}}\ \text{x}\ 100\%$$

$$= \frac{|3800-3800|}{3800} \text{ x } 100\ \%$$

$$= 0\%$$

Dan terjadi *overshoot* dengan nilai sebesar:

$$\%\ \text{Mp} = \frac{|\text{RPMpuncak} - \text{RPM } steady|}{\text{RPM } steady} \text{ x } 100\ \%$$

$$= \frac{|3945-3800|}{3800} \text{ x } 100\ \%$$

$$= 3{,}816\%$$

Pada saat kenaikan *set point* dari 3800 rpm menuju 4440 rpm, respon memiliki % *error* sebagai berikut:

$$\%\ Ess = \frac{|Average\ \ speed\ \ steady - \text{setpoint}|}{\text{setpoin t}} \text{ x } 100\%$$

$$= \frac{|4440-4440|}{4440} \text{ x } 100\ \%$$

$$= 0\%$$

Dan terjadi *overshoot* dengan nilai sebesar:

$$\%\ \text{Mp} = \frac{|\text{RPMpuncak} - \text{RPM } steady|}{\text{RPM } steady} \text{ x } 100\ \%$$

$$= \frac{|4464-4440|}{4440} \text{ x } 100\ \%$$

$$= 0{,}54\%$$

Pada saat penurunan *set point* dari 4440 rpm menuju 4000 rpm, respon memiliki % *error* sebagai berikut:

$$\%\ Ess = \frac{|Average\ \ speed\ \ steady - \text{setpoint}|}{\text{setpoint}} \text{ x } 100$$

$$= \frac{|4000-4000|}{4000} \text{ x } 100\ \%$$

= 0%

Dan terjadi *overshoot* dengan nilai sebesar:

$$\% \text{ Mp} = \frac{|\text{RPMpuncak} - \text{RPM } steady|}{\text{RPM } steady} \text{ x } 100\ \%$$

$$= \frac{|3983 - 4000|}{4000} \text{ x } 100\ \%$$

$$= 0{,}425\%$$

Pada saat kenaikan *set point* dari 4000 rpm menuju 4200 rpm, respon memiliki % *error* sebagai berikut:

$$\% \textit{Ess} = \frac{|Average\ \ speed\ \ steady - \text{setpoi nt}|}{\text{setpoint}} \text{ x } 100$$

$$= \frac{|4200 - 4200|}{4200} \text{ x } 100\ \%$$

$$= 0\%$$

Dan terjadi *overshoot* dengan nilai sebesar:

$$\% \text{ Mp} = \frac{|\text{RPMpuncak} - \text{RPM } steady|}{\text{RPM } steady} \text{ x } 100\ \%$$

$$= \frac{|4207 - 4200|}{4200} \text{ x } 100\ \%$$

$$= 0{,}17\%$$

Dari grafik Gambar 5.15 terlihat bahwa ketika terjadi perubahan *set point*, nilai *overshoot* dari respon sistem semakin lama semakin kecil, dan nilainya tidak melebihi kriteria *overshoot* sebesar 5%. Serta nilai *error steady state* (ess) sebesar 0% disetiap nilai *set point*.